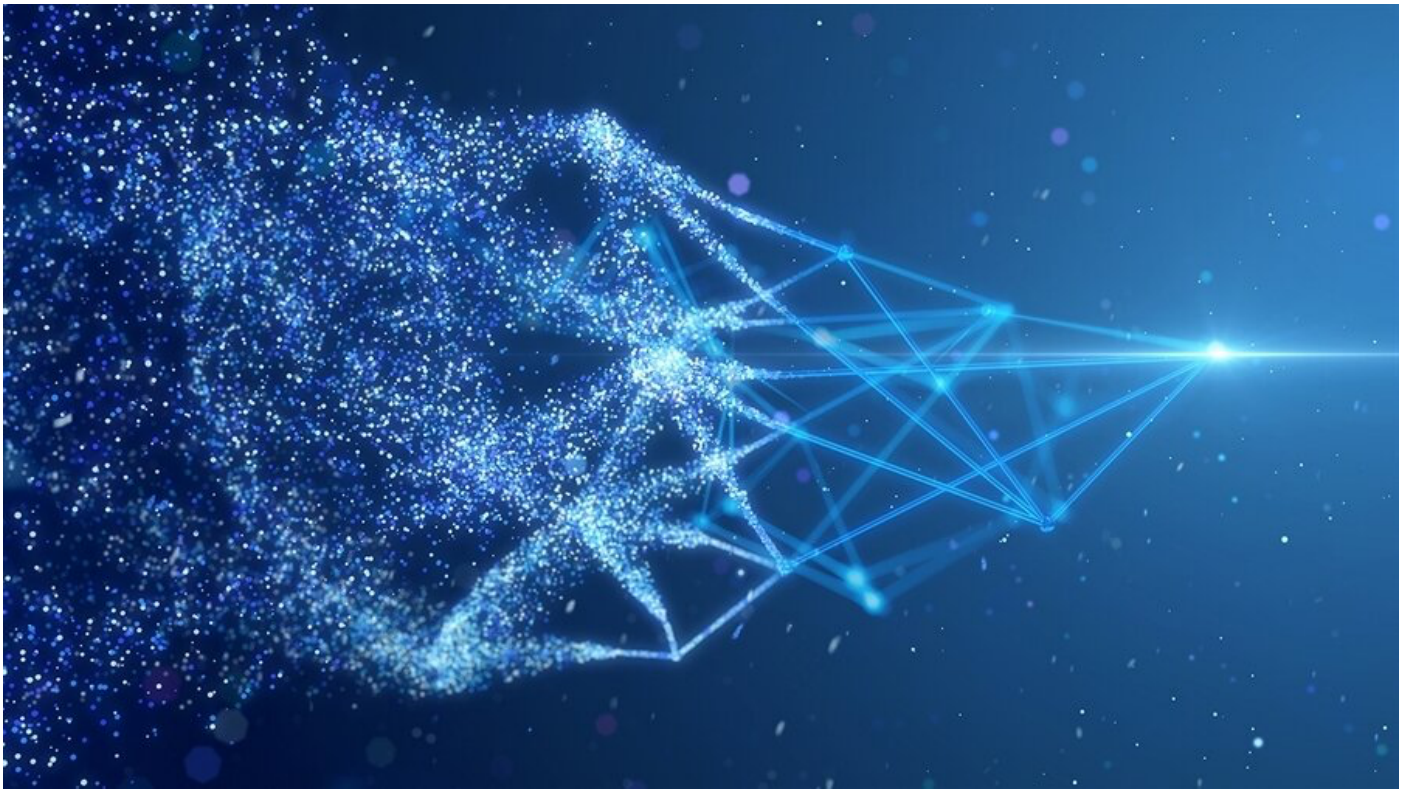
Container Runtimes and WasmEdge benchmarking on Arm



Howard Zhang
May 17, 2023

6 minute read time.

In this blog, I would like to introduce some runtimes on Arm64 platform. Especially, I give you a benchmarking on WasmEdge and Runc, to show you the advantage of Wasm compared with container. Arm consistently makes

contribution to open-source projects. We actively participate in the following cloud-native projects that focus on enabling, optimizing performance, and implementing continuous integration and deployment on the Arm64 platform.

Runtimes

A Container runtime is software that is responsible for running containers. Runc is a widely applied runtime and is used by Containerd and Docker by default. There are some other runtimes, such as Crun, Runsc, and Kata containers. Here is the responsibility of a runtime:

- Consuming the container mount point
- Consuming the container metadata (can also be a manually crafted config.json for testing)
- Communicating with the kernel to start containerized processes (clone system call)
- Setting up cgroups, SELinux Policy, App Armor rules

Each runtime has its own adapted areas, like lightweight applications or a Secure environment.

Kata Container

Kata Container is a secure container runtime that uses hardware virtualization technology as an additional layer of defense. This allows it to provide stronger workload isolation while still being lightweight and performing like regular containers.

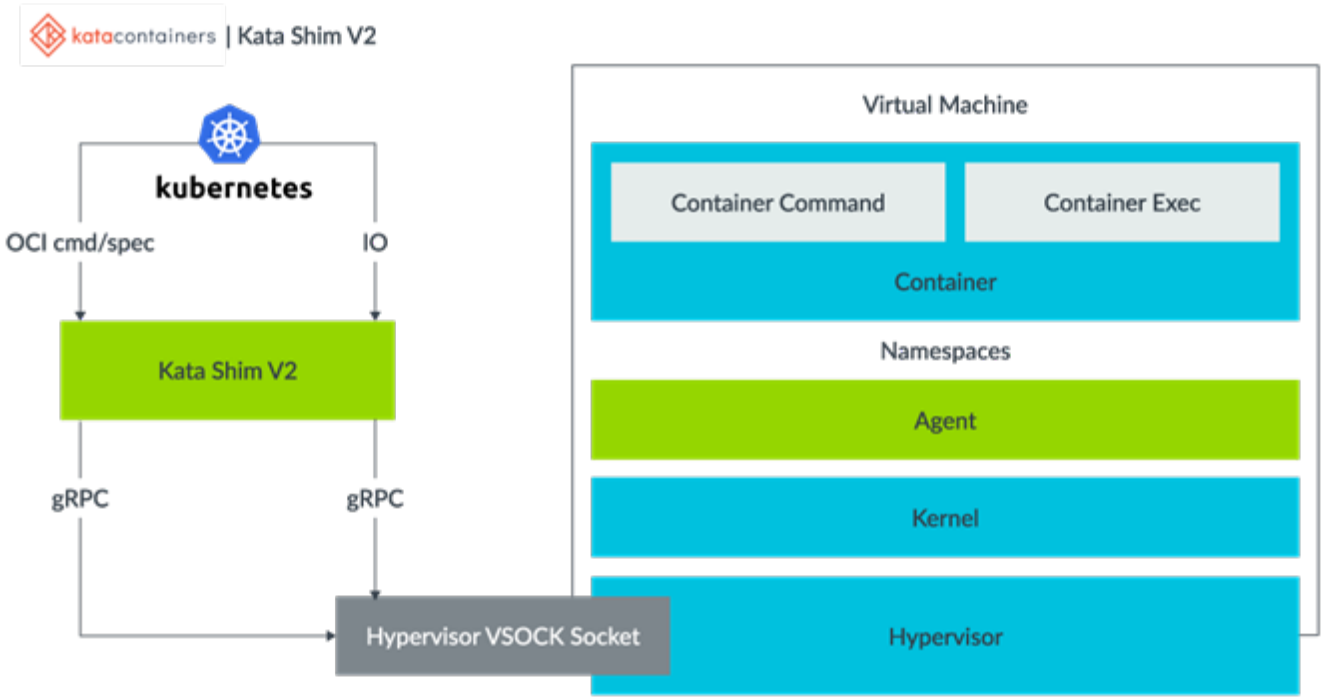


Image source : https://katacontainers.io/learn/

Moreover, we change the default Virtual Machine Monitor (VMM) from Qemu to Cloud Hypervisor. Compared to Qemu, Cloud Hypervisor is more lightweight and Secure. It is because Cloud Hypervisor has minimal emulated devices and is implemented in Rust to avoid many common security issues.

For more details, please refer to https://katacontainers.io/ and https://www.cloudhypervisor.org/.

gVisor

gVisor is an application kernel, written in Go, that implements a substantial portion of the Linux system call interface. It provides an extra layer of isolation between running applications and the host operating system. It takes advantage of KVM to keep the isolation of memory and CPU between

container and host. It also has an application kernel that handles most syscalls in user space, and only a limited number of syscalls would pass to the host kernel. This reduces the attack surface.

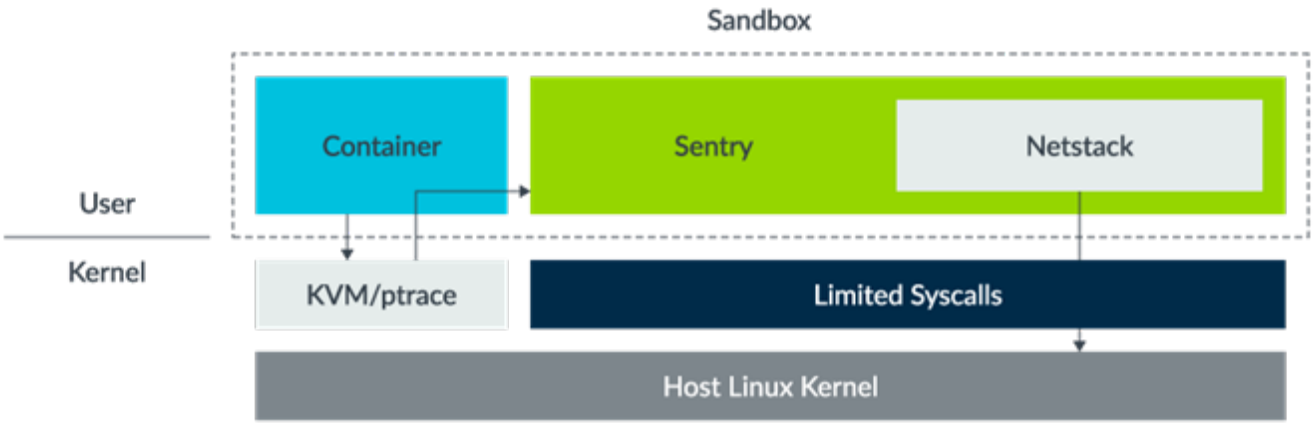


Image Source: https://gvisor.dev/blog/2020/04/02/gvisor-networking-security/

For more details, please refer to https://gvisor.dev/.

WasmEdge

WasmEdge is a lightweight, high-performance, and extensible WebAssembly runtime for cloud-native, edge, and decentralized applications.

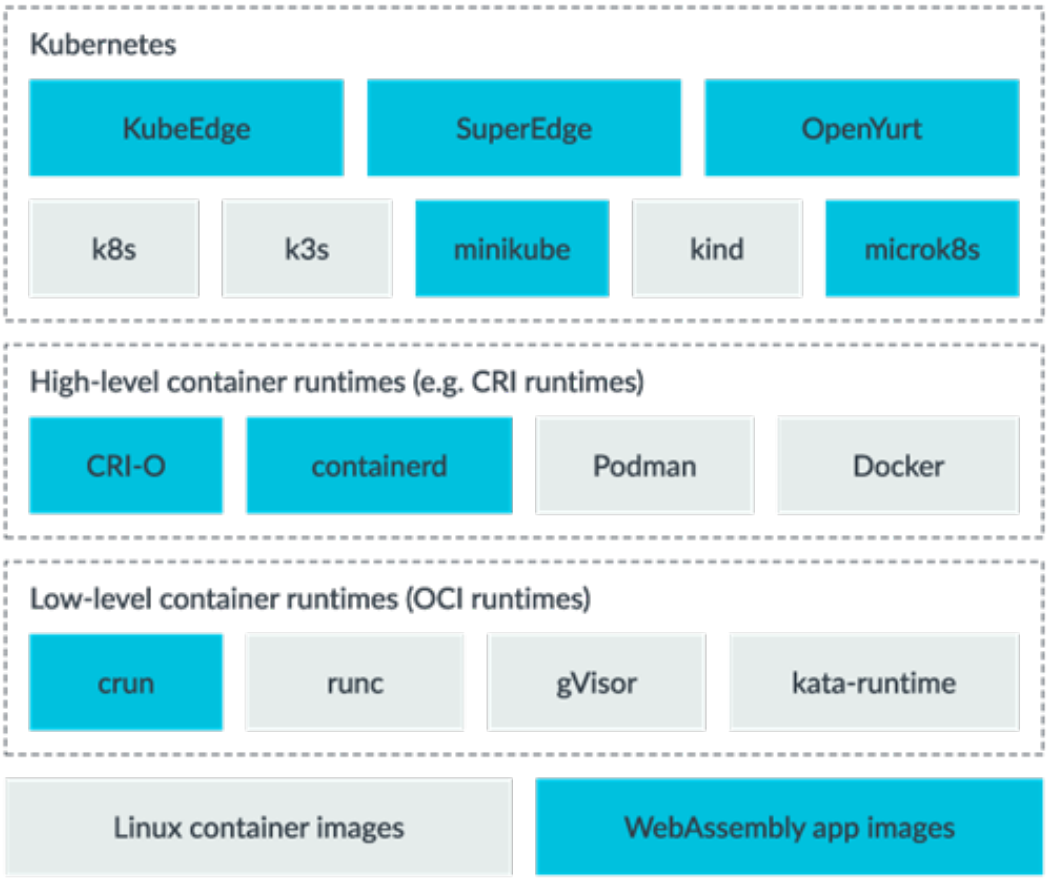


Image source: https://wasmedge.org/book/en/use_cases/kubernetes.html

WasmEdge uses WebAssembly and WASI. I would not go deep into these two technologies but a general introduction.

- WebAssembly (abbreviated Wasm) is a binary instruction format for a stack-based virtual machine.
- A memory-safe, sandboxed execution environment
- Program languages, like rust, c and Golang, can be compiled into the Wasm binary format that is compatible across all these platforms
- WASI is a modular system interface for WebAssembly. Wasm app interacts with a host kernel through WASI.

For more details, please refer to https://wasmedge.org/book/en/, https://github.com/WebAssembly/WASI, and https://webassembly.org/.

Generally speaking, compared with container technology, Wasm has several pros.

- Smaller binary size and memory footprint
    - Some libraries can be shared between Wasm binary.
- Faster start time
    - Do not have some startup works in container technology, like creating namespace, cgroup, and so forth.
- Less relay on host kernel
    - Container technology needs namespace, cgroup, filesystem, and so forth, but Wasm does not need them.

However, as an unmatured technology, it does have some drawbacks:

- WASI only supports limited syscalls, many applications are not able to migrate to Wasm
- Security of it still needs to be verified
- Many import features are still unimplied. https://github.com/WebAssembly/proposals

These advantages make Wasm quite fit for Serverless and edge computing, which is sensitive to image size or startup time.

A micro-benchmark between WasmEdge and Runc

We do three benchmarks to illustrate the advantage between WasmEdge and Runc, all benchmarks are done on Arm64 server. Here is some information about the testing platform.

CPU	Ampere(TM) Altra(TM) Processor

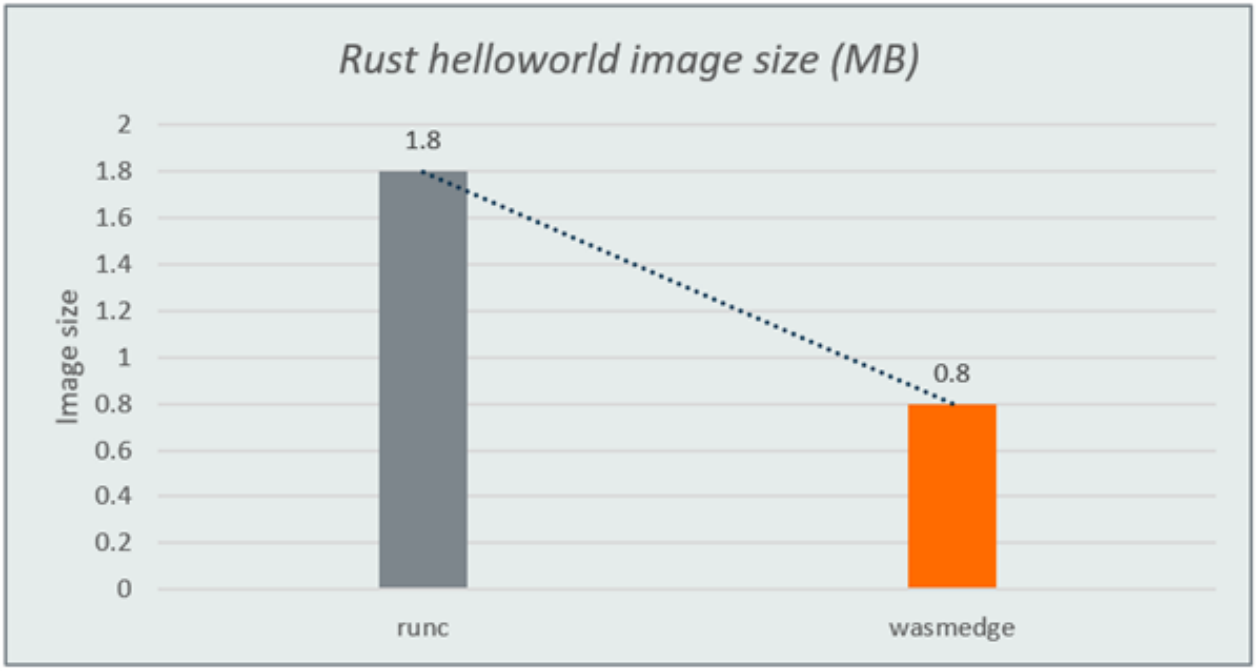
kernel	5.17.0
runc	1.1.4
WasmEdge	0.11.0

The first test verifies the size between normal rust binary (it is used by normal container) and Wasm binary. The following is the code.

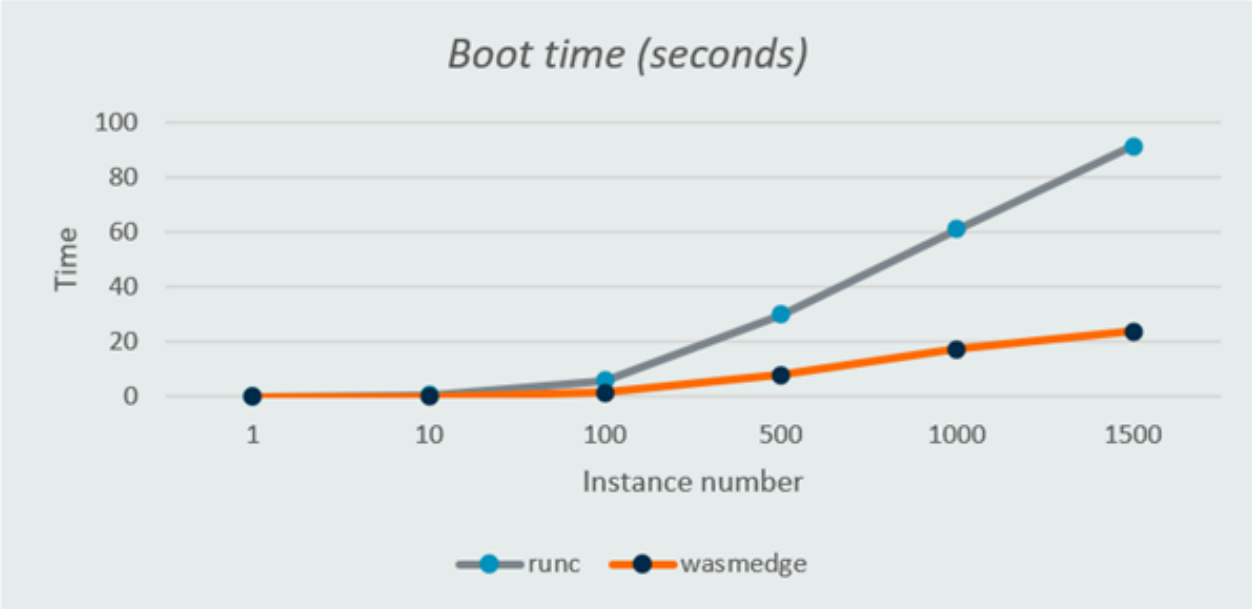
Fullscreen

```
1  fn main() {
2
3    println!("hello");
4
5  }
```

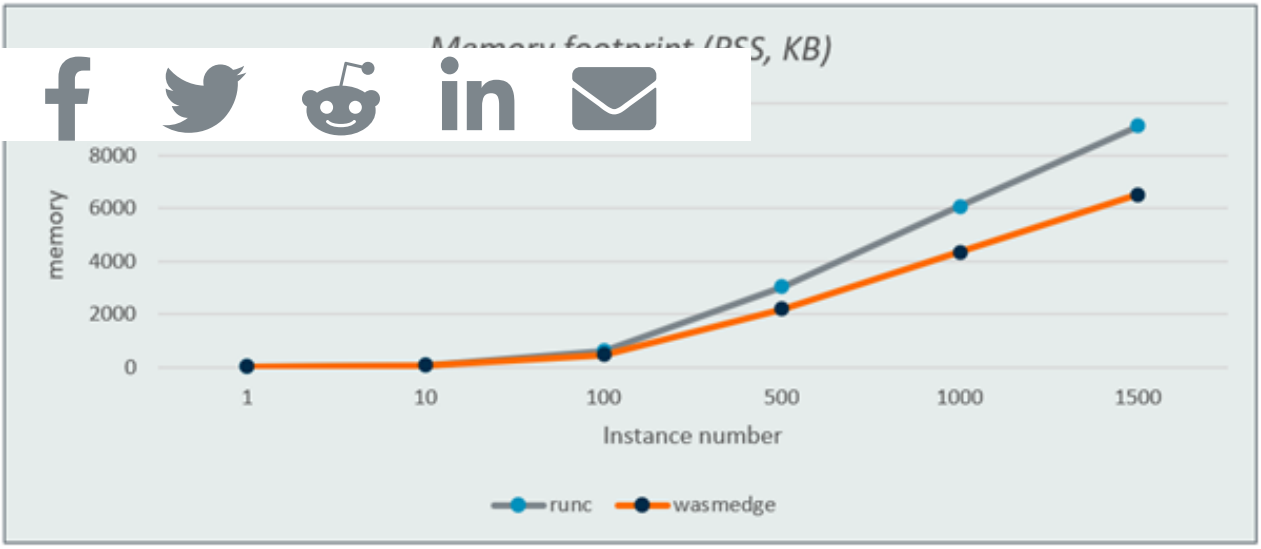
The normal rust binary is 1.8MB while the Wasm binary is only 0.8MB.



The second test is repeat start the helloworld image that is, runc and WasmEdge for 1, 10, 100, 500, 1000 and 1500 times. As you can see in the second image, the boot time for Wasm is around 25% of that for normal container.



The third test is runs in parallel by starting several helloworld images and making them sleep for a while, then count the memory usage. As you can see in the third image, the memory footprint of 1500 Wasm is 25% less than that of 1500 normal containers.



Summary

In this blog, we give you a general introduction of runtimes, and shows you

the benchmark of image size, boot time and memory footprint between runc and WasmEdge.

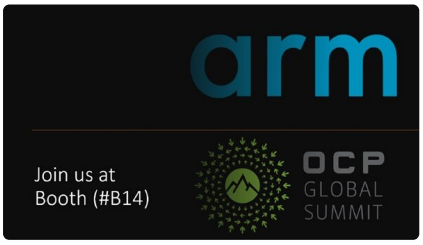
Here is a table for summaries runtimes in following dimensions, image size, boot time, security, difficulty of app development, project maturity and adapted environment to evaluate.

	runc	Kata/gVisor	WasmEdge
image size	normal	normal	small
Bootup time	normal	slow	fast
Security	normal	high	unknown
Difficulty of App development	normal	normal (debugging within the container is a problem)	high
Project maturity	high	high	low
adapted environment	Normal usage	Strong isolation needed	Boot time or memory size sensitive

If you are interested in these runtimes, you can visit their official websites to obtain more detailed information.

WasmEdge: https://wasmedge.org/
WebAssembly: https://webassembly.org/.
Kata Containers: https://katacontainers.io/
Cloud Hypervisor: https://www.cloudhypervisor.org/.
gVisor:https://gvisor.dev/.

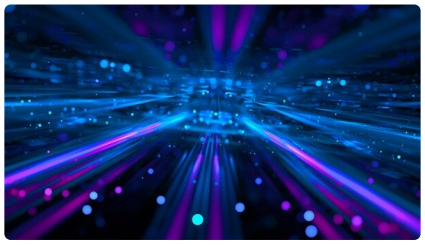
Infrastructure Solutions blog



Join Arm and Our Ecosystem at OCP Global Summit 2023

Arm is excited to be back at the OCP Global Summit 2023 year to showcase the Arm ecosystem's innovation and collaboration in open compute.

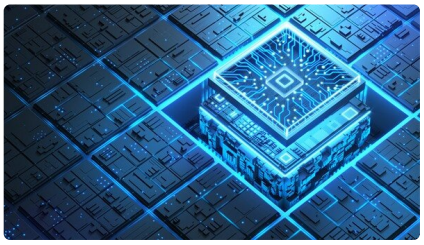

Imran Yusuf



Nginx Performance on AWS Graviton3

In this blog we explore the performance of an Nginx Reverse Proxy (RP) and API Gateway (APIGW) on the AWS Graviton3 compared to 6th generation AWS EC2 instances.


Julio Suarez



Neoverse Compute Subsystems, the Fastest Path to Production Silicon

We talk about a new product offering, Arm Neoverse Compute Subsystems (CSS) and the first product, Arm Neoverse CSS N2.


Steve Demski

More blogs in Arm Community blogs

AI and ML blog

Announcements

Architectures and Processors blog

Automotive blog

Embedded blog

Graphics, Gaming, and VR blog

High Performance Computing (HPC) blog

Infrastructure Solutions blog

Internet of Things (IoT) blog

Operating Systems blog

SoC Design and Simulation blog

Tools, Software and IDEs blog

Tags

- Open Source Projects
- containers
- Cloud Native

Actions

Related blog posts

TensorFlow and PyTorch container monthly releases by Arm

We are happy to announce monthly releases of TensorFlow and PyTorch container images for Arm servers,...

TensorFlow and PyTorch container monthly releases by Arm

We are happy to announce monthly releases of TensorFlow and PyTorch container images for Arm servers,...

ARM Powered Smartphone Sets New Graphics Benchmark

Taking an in-depth look at all methods used throughout the industry to compare GPU performance in the handset s...

ARM Powered Smartphone Sets New Graphics Benchmark

Taking an in-depth look at all methods used throughout the industry to compare GPU performance in the handset s...

Arm Allinea Studio 19.3: performance improvements in preparation for SC'19 benchmarks

Arm Allinea Studio 19.3 is now available . This release adds major new features for our customers to test a...